IN THE UNITED STATES

PATENT AND TRADEMARK OFFICE

APPLICATION OF MARTIN G. REIFFIN FILED: SEPTEMBER 28, 1982 FOR: COMPUTER SYSTEM WITH REAL-TIME LANGUAGE PROCESSING SERIAL NO. 425,612	ART UNIT 232 Examiner T. Lee

AFFIDAVIT of LAWRENCE OPPENHEIM

County of San Francisco)
) ss.
State of California)

LAWRENCE OPPENHEIM, being duly sworn, deposes and says that:

1. I reside at 259 Marietta Drive, San Francisco, California 94127.

2. I received the degree of Bachelor of Arts in English Literature from Pomona College in 1971 and the degree of Master of Library Science from the University of Southern California in 1975.

3. I have completed both elementary and advanced college courses in each of the following programming languages: Data Base, COBOL, IBM 370 Assembler, and Pascal.

4. I have been employed as an applications and systems programmer by the the following employers for the periods indicated:

1978-1979: Southern Pacific Railroad, SanFrancisco.

1979-1981: Crum & Foster Insurance Co., San Francisco.

1981-1982: Federal Reserve Bank of San Francisco.

1982-1983: VWR Scientific, Brisbane, CA.

1983-1985: Bank of America, San Francisco.

5. From 1985 to the present I have been an independent consultant doing business as Penguin Computing at 15 West Portal, San Francisco. My services consisted of systems programming and applications programming for both mainframe computers and IBM personal computers, and the design and set up of computer systems for businesses.

6. Since 1986 I have been teaching computer science courses at the University Extension of the University of California at Berkeley, including courses in Pascal, IBM PC Assembler Language, EXEC-2 AND REXX. I am presently teaching IBM PC Assembler Language at that university for the second time. I have also taught courses in C Language and REXX for employees of Bank of America and VMAssist Inc., respectively, in San Francisco.

7. The course in IBM PC Assembler Language which I have taught previously and am now teaching again at the University of California is described in the University catalog as follows:

"This course provides an introduction to 8088 assembler language, the most direct way to control every aspect of the IBM PC. It is intended for PC users and programmers who would like a deeper understanding of what is 'under the hood'.

***

"Topics include the 8088's segmented memory, registers, instruction set, and addressing modes; the IBM PC ROM BIOS; common DOS function calls; how to interface with higher-level languages; and the difference between COM and EXE files. Finally, the instructor illustrates how to use DEBUG to 'disassemble' and optimize code produced by Turbo Pascal."

8. I have read and studied the Office Action dated 06/29/89 in the above application, the Lawrence et al. patent 4,464,730, the Maddock patent 4,513,391, the Atkinson et al. publication entitled "COPAS --A Conversational Pascal System", Claims 38 to 58 and 64 to 66 inclusive in the Amendment dated February 28, 1989 and the Supplemental Amendment dated March 9, 1989, and the specification and drawings in the above application.

9. The Office Action dated June 29, 1989 contains the following inaccurate assertions allegedly describing the operation of the systems disclosed in the Lawrence et al. and Maddock patents:

(1) That the editor of these patents is an "interrupt service routine" (Paragraphs 16, 20, 22 and 28);

(2) That "when a key was struck, the reformatting process was interrupted immediately and the control of the system was transferred to the editor" (Paragraph 22);

(3) That "After the reformatter finished the reformatting process, the reformatter still had the control of the central processing unit until it was interrupted" and "it went into a loop" (Paragraph 28);

(4) That the Lawrence et al. formatter is a "scanner" and a "parser" (Paragraph 32);

(5) That "a routineer in the DP art would have recognized that the language analyzer taught by Lawrence et al. and Maddock would have been exchangeable with the language

analyzer taught by Atkinson et al." (Paragraph 13); and

(6) That "Atkinson et al's language analyzing means would have increased the versatility of Lawrence et al and Maddock's system" (Paragraph 13).

10. The Lawrence et al. and Maddock patents do not disclose that their editor is within an "interrupt service routine". On the contrary, the Maddock patent clearly reveals that this alleged arrangement is literally impossible in the system disclosed in this patent, as explained below.

11. The Examiner's reasoning appears to be based upon a misconception that the mere disclosure in Maddock of a keyboard actuated interrupt establishes that the editor is, or is within, an interrupt service routine. This is erroneous. This IBM patent discloses nothing to distinguish its keyboard interrupt system from the conventional IBM type-ahead keyboard interrupt system that is present in every one of the millions of IBM personal computers and their clones.

12. The type-ahead keyboard interrupt routine of the IBM personal computer merely places a character code in the buffer queue in response to striking a key. This keyboard interrupt routine is not an "editor" and is not called an "editor" by IBM or anyone else. As shown in the attached Exhibit A showing Page A-29 of the listing of the ROM BIOS constituting Appendix A of the IBM Personal Computer Hardware Reference Library Technical Reference, this routine is called "Keyboard Interrupt Routine".

13. All of the hundreds of different editors and word

processors for the IBM personal computer make use of this keyboard interrupt service routine, either directly or through the operating system, and none of these editors or word processors is regarded by those skilled in the art or in the literature as constituting an interrupt service routine. The keyboard interrupt service routine listed in Exhibit A does not perform the functions of an editor. After this interrupt service routine places each character code in the buffer it returns control of the central processing unit to the interrupted program. The interrupt service routine is then terminated. Thereafter, whenever the editor is ready for the next character or command, the editor may remove one or more character codes from the buffer and perform the appropriate editing actions, but this is done only after the keyboard interrupt service routine has ended and returned control of the central processing unit to the interrupted program.

14. The Examiner's assumption that the alleged presence of a keyboard interrupt service routine in the referenced IBM patents establishes that the editor constitutes an "interrupt service routine" or gets control of the central processing unit immediately in response to an interrupt, is erroneous.

15. The following description of the type-ahead keyboard interrupt system of the IBM personal computer will make it clear that this type-ahead system, like those in the IBM reference patents to Maddock and Lawrence et al., does not include the editor within the keyboard interrupt service routine.

16. Upon studying the Lawrence et al. patent one skilled in the art would have no reason to believe that the keyboard

interrupt system of this IBM patent operates differently from that of the IBM personal computer, as described summarily above and in more detail below. Upon studying the Maddock patent one skilled in the art would infer that the keyboard interrupt system of this IBM patent does not operate differently from that of the IBM Personal Computer as described summarily above and in more detail below, and he would understand that the disclosed Maddock system does not include the editor within the interrupt service routine, for the reasons stated below.

17. The type-ahead keyboard interrupt system of the IBM personal computer is disclosed in the copied publication pages attached hereto as the following exhibits:

Exhibit B: "8088 Assembler Language Programming: The IBM PC", D. C. Willen and J. L. Krantz, Pages 92-101, Howard Sams & Co., 1983.

Exhibit C: "Programmer's Problem Solver for the IBM PC, XT. & AT", R. Jourdain, Pages 89-100, 109-111, Brady Commun. Co., Inc., 1986.

Exhibit D: "MS-DOS Developer's Guide", The Waite Group, Pages 194-198, Howard Sams & Co., 1989.

Exhibit E: "Assembly Language Programming for the IBM Personal Computer", D. J. Bradley, Pages 235-239, Prentice-Hall, Inc., 1984.

18. As shown in Exhibits B, C, D and E, the IBM type-ahead keyboard interrupt system of the IBM Personal Computer operates as follows:

(1) At the hardware level supporting the operating system MS-DOS is a scheme of hardware interrupts, each of which is associated with a particular device. Each device wanting the processor's attention sends an interrupt request, or IRQ, to the 8259 interrupt controller which schedules the interrupt for service. The 8259 can handle up to eight interrupting devices. The IBM Personal Computer connects the system timer, keyboard, asynchronous communications adapter, fixed disk, diskette, and printer to interrupt lines. The remaining interrupt lines are available for use by other I/O devices installed in the system I/O channel. The hardware design assigns each of these interrupting devices to a particlar interrupt input on the 8259 interrupt controller. When the latter decides that an interrupt can be serviced, it sends an "interrupt acknowledge" message to the device, disables all further interrupts, and generates an interrupt process.

(2) In response to a particular hardware interrupt, the central processing unit looks up the address of an interrupt service routine in the interrupt vector table stored in low memory. The processor pushes the flag register and the code segment and instruction pointer registers onto the stack and begins executing the interrupt service routine located at the memory address obtained from the interrupt vector table.

(3) After the interrupt service routine performs its function it sends an end of interrupt (EOI) message to the 8259 controller, indicating that it is ready to accept another interrupt service request. The interrupt service routine finally executes an IRET instruction that restores the flags and the original values of the code segment and instruction pointer

registers. The program which was interrupted will then resume control of the central processing unit at the very next machine code instruction following the instruction which was being executed at the instant of the interrupt.

(4) The keyboard contains an 8048 microprocessor which senses each keystroke and deposits a scan code in Port A of the 8255 peripheral interface chip located on the system board. A scan code is a one-byte number in which the low seven bits represent an arbitrary identification number assigned to each key. After each keystroke the 8255 issues an "acknowledge" signal back to the microprocessor in the keyboard.

(5) When the scan code is deposited in Port A, the keyboard interrupt (INT 9) is activated. If the 8259 interrupt controller determines that a higher-priority interrupt is not in service, the controller activates the keyboard interrupt line to the central processing unit. Immediately upon completion by the central processing unit of the execution of the current machine code instruction, the flag, code segment and instruction pointer registers are pushed onto the stack, and the central processing unit executes an interrupt acknowledge cycle. The 8259 responds to the cycle with the interrupt number 9. The central processing unit immediately fetches the CS:IP (code segment:instruction pointer) memory address of the interrupt service routine from memory locations 24H and 26h of the interrupt vector table and begins execution of the interrupt service routine at that address.

(6) The INT 9 interrupt service routine translates the scan code into an ASCII character code by accessing a data table

with the XLAT instruction. The routine then checks to see if there is any room in the buffer for another character code. The buffer is constructed as a circular queue. New character codes are inserted at successively higher memory positions in the buffer, and when the upper limit is reached, the insertion wraps around to the low end of the buffer where additional codes may be stored until the buffer is full. Up to fifteen character codes may be present in the buffer at any one time.

(7) When the INT 9 interrupt service is finished the interrupt service routine sends an EOI command to the 8259. This resets the latter so that any lower-priority interrupts waiting for service may now be acknowledged and processed. At the very end of the interrupt service routine it executes the IRET instruction which pops the stack to restore the IP (instruction pointer), CS (code segment) and flag registers to their values prior to the interrupt.

(8) After the keyboard interrupt service routine has terminated and returned control of the central processing unit to the program which was interrupted by the keystroke, the editor (or word processor or other program) may obtain a character code from the buffer either by calling directly a subroutine ("software interrupt") in the BIOS (Basic Input Output System) or by calling a subroutine in the operating system which in turn invokes the BIOS routine. These subroutines remove a character code from the buffer queue and pass the code to the editor or other calling program which invoked them.

19. It is important to note that in the IBM Personal Computer keyboard interrupt system the INT 9 keyboard interrupt routine occurs asynchronously with respect to the main program

(e.g. an editor or word processor) running in the computer. What this means is that the striking of a key and execution of the keyboard interrupt routine can occur at any time, and it is totally independent of when the main program may wish to receive keyboard input. Conversely, the editor or word processor may obtain a character code from the buffer queue whenever the editor is ready to receive the code. Therefore a plurality of character codes may be stored in the queue at any one time before the editor (or other program) gets around to removing any of them from the queue. This is the very reason why the buffer is in the form of a queue.

20. Because of the above-described asynchronous relation between the INT 9 interrupt service routine and the editor's acquisition of the character codes from the buffer queue, the queue may have as many as fifteen character codes stored therein at any one time. The Maddock specification discloses at Col. 4, Lines 20-27 that he also stores a plurality of character codes in the buffer queue at the same time. The very meaning of the term "queue" makes this clear. Therefore the keyboard interrupt service routine of the IBM Maddock patent must also be asynchronous with respect to the editor.

21. That is, the Maddock editor cannot possibly be within the interrupt service routine. If the Maddock editor were within the interrupt service routine so as to be activated immediately in response to each and every keystroke, each new character code would be immediately removed from the queue by the editor during the interrupt service routine. There would then be a synchronous relation between the insertion of a character code into the buffer and the removal of that code from the buffer by

the editor. As a result, there would never be more than a single character code in the queue at any time, and the buffer would not be a "queue" capable of storing a plurality of characters. Only the asynchronous operation of the IBM keyboard interrupt described above with respect to the IBM Personal Computer could result in the accumulation of a plurality of character codes in the buffer queue, and this is what Maddock expressly discloses. This asynchronous relation between the Maddock editor and his keyboard interrupt service routine precludes any possibility that the Maddock editor "is an interrupt service routine" as erroneously asserted by the Examiner.

22. Furthermore, the Maddock specification states at Col. 5, Lines 52-54 that:

"...once the background formatting is commenced it will not be interrupted until the reformatting of the current line is completed." (Emphasis added.)

An interrupt service routine, by definition, gets control immediately upon completion of the machine code instruction being executed at the instant of the interrupt. This is a period measured in microseconds. An interrupt service routine cannot be delayed "until the reformatting of the current line is completed." The only circumstance that can delay passing control to the interrupt service routine is the present activation of a different interrupt having a higher priority. This circumstance is not involved in the Maddock system.

23. In the Maddock patent disclosure the editor and formatter constitute two coroutines. As shown in Fig. 3 of the Maddock patent each coroutine relinquishes control to the other coroutine voluntarily. This means that neither coroutine is interrupted by or interrupts the other coroutine. Therefore

neither the editor nor the formatter can be regarded as an interrupt service routine.

24. Paragraph 20 of the Office Action erroneously relies upon the following statement in Col. 4, Lines 20-22 of the Maddock specification in support of the Examiner's contention that the Maddock editor is an interrupt service routine which gets control immediately upon striking a key:

"...during text editing operations, keystrokes from the keyboard 8 are processed by an interrupt handler 14 forming part of the editor/formatter software..."

That the author of the Maddock specification chose to label all of the software, including the interrupt handler, with the designation "editor/formatter software" does not mean that the editor portion of this software is an interrupt service routine which gets control immediately upon striking a key. If the Examiner's logic and interpretation of this statement were correct then the formatter would also be an "interrupt service routine", in which case the Examiner's inaccurate contention that the formatter operates continuously in the background would be further refuted.

25. The IBM patent to Lawrence et al. discloses neither the hardware, software nor mode of operation of the alleged keyboard interrupt routine. However there is nothing in this disclosure which is inconsistent with that of the IBM personal computer type-ahead keyboard system described above or which would suggest to one skilled in the art that in the Lawrence keyboard interrupt system the editor "is an interrupt service routine".

26. To modify the Lawrence et al. or Maddock system by

including the editor within a hardware interrupt service routine would not have been obvious in 1982 because it would have been pointless, would have been alien to the conventional practise, would have involved needless complication in the implementation without any compensating benefit or advantage, and is not suggested by any prior art relied upon by the Examiner in said Office Action. Since the time required to format characters on a monitor screen is so brief as to be almost instantaneous there would have been no need or purpose in such a modification of the reference disclosures. Only in applicant's language analyzer system where lexical, syntactic and semantic analyses may take a time period of many orders of magnitude longer than a screen formatting operation does the inclusion of the editor wihin an interrupt service routine become meaningful and advantageous. That is, it allows applicant's language analyzer to run constantly in the background and to maintain control of the central processing unit for almost the entire time without repeatedly polling the operating system to inquire if a keystroke has arrived for processing by the editor.

27. The Examiner's assertion in Paragraph 22 of the Office Action that in the Maddock disclosure "when a key was struck, the reformatting process was interrupted immediately" is erroneous. As pointed out above in Paragraph 16, this assertion is directly contrary to Maddock's specification at Col. 5, Lines 52-54, unless by "immediately" the Examiner means something different than that understood by those skilled in the art when discussing the activation of an interrupt service routine.

28. The Examiner's assertions in Paragraph 28 of the Office Action that "After the reformatter finished the

reformatting process, the reformatter still had the control of the central processing unit until it was interrupted" and that "it went into a loop" are unsupported by any disclosure in the Maddock patent. I am unable to find any mention or hint of such a "loop" in this patent. Furthermore, I can conceive of no reason or purpose for such a mode of operation in the Maddock system since there would be no point in having a subroutine maintain control of the system after the subroutine has completed its function.

29. I do not understand the Examiner's assertion in Paragraph 13 of the Office Action that "a routineer in the DP art would have recognized that the language analyzer taught by Lawrence et al. and Maddock would have been exchangeable with the language analyzer taught by Atkinson et al." If the Examiner means that it would have been obvious to a routineer to substitute Atkinson's emasculated beginner's compiler for the formatter of Lawrence et al. or Maddock then the Examiner's assertion is erroneous for several reasons stated below:

(a) First, the substitution of Atkinson's compiler for the formatter of the reference patents would have rendered the patented systems useless for their intended purpose. Lawrence et al. and Maddock disclose the IBM 3730 system comprising the IBM 3791 controller and IBM 3732 display terminal. This system is a general-purpose screen editing system intended for professional use and required to handle substantially all languages. As opposed to this, the Atkinson compiler is "intended for novices", does not even support the whole Pascal language, and is incapable of handling anything but "short programs" written by students. These limitations of the Atkinson

compiler are inherent and cannot be obviated because it is an incremental compiler, as explained below. The substitution of Atkinson's novice emasculated-Pascal compiler for the formatter in the IBM 3730 terminal system would merely destroy the utility of an expensive complex IBM system. The complexity and expense of the IBM 3730 system disclosed in the reference patents can be justified only by the broad general utility of the system for handling all text editing work for all languages, both natural and formal. To convert such complex expensive hardware into a student compiler for a single emasculated language would be pointless, and any such suggestion made to one skilled in the art would be regarded as ill advised. The Examiner's statement in Paragraph 13 of the Office Action that "Atkinson et al's language analyzing means would have increased the versatility of Lawrence et al and Maddock's system" is untenable. To modify the IBM system so that it cannot perform its intended general-purpose screen editing function would be neither obvious nor properly characterized as "increased versatility".

(b) Second, the IBM screen editor hardware is in an art that has different functions, structures and problems than the compiler software art of the Atkinson publication. Also, the Atkinson student compiler is "implemented for hard-copy terminals". With respect to both structure and mode of operation a hard-copy terminal is very different from a screen terminal. Therefore to one skilled in the art of screen editing hardware the Atkinson compiler software would have no apparent relevance. If he wished to improve his screen editing hardware he would be unlikely to turn to the prior art of compiler software, and even less likely to turn to compiler software implemented for a hard-copy terminal.

(c) Third, the Atkinson incremental compiler is an illustration of the very problem to which the Reiffin invention is directed, not the solution. After entry of even the slightest change in the source code the programmer using the Atkinson compiler must sit and wait until the Atkinson compiler recompiles the entire code from the beginning. During this waiting period he cannot enter a single keystroke. For any but small programs written by novices this waiting period would be too lengthy to be tolerated. Therefore the Atkinson incremental compiler scheme is not suitable for professional use, and would be totally unacceptable in professional equipment such as the IBM systems disclosed in the reference patents.

30. The assertion in Paragraph 32 of the Office Action that the Lawrence et al. formatter is a "scanner" and " parser" is inaccurate, for the reasons stated below:

(1) A "scanner" performs a lexical analysis of text. That is, it determines if each sequence of text characters constitutes a properly spelled word or token. I can find no disclosure in the Lawrence et al. patent of such a function.

(2) A "parser" performs a syntactic analysis of text. That is, it determines if a sequence of words or tokens conforms to the grammatical rules of either a formal or natural language. I can find no disclosure in the Lawrence et al. patent of such a function.

31. From the standpoint of one skilled in the compiler art and desirous of improving the Atkinson compiler, it would not be obvious to "exchange" the Atkinson compiler for the Lawrence

et al. or Maddock formatters as proposed by the Examiner at Page 4, Paragraph 13 of the Office Action dated June 29, 1989. Such an exchange would be futile and pointless because no improvement in the performance of the Atkinson compiler could result from such a substitution, for the reaons stated below.

32. The Atkinson compiler is incremental. That is, it analyzes the syntax only of complete declarations or statements each of which must be fully set forth in a single line. If the programmer attempted to spread a statement over two or more lines, as permitted in every popular Pascal compiler of which I am aware, the Atkinson compiler would stop and emit an error message upon completing its syntax analysis of the incomplete statement in the first line, since an incomplete statement is grammatically incorrect to an incremental compiler.

33. Furthermore, as stated at Page 821, Line 6 of the Atkinson et al. publication:

"Input is buffered, transmission of a buffer being effected by the RETURN key."

That is, the compiler cannot even get access to the line of source code until the line is complete and entered. Therefore the Atkinson incremental compiler must wait until each line of source code has been completed and entered before the compiler can even begin to process the line. While the programmer is typing the line at the keyboard the Atkinson compiler can do nothing but wait until it receives a signal (the pressing of the carriage return key) that the complete line has been taken from the buffer and entered. During this waiting period, the Atkinson compiler is idle and performs no useful work.

34. For example, in the penultimate line on Page 824 of

the Atkinson et al. publication the following line is typed in during the demonstration of the compiler:

"20 writeln (n,'squared is', sqr(n))"

This line contains an intentional error in that the variable "n" has not been declared. The compiler does not analyze this statement and recognize the error as soon as the variable "n" is typed immediately after the first open parenthesis. It is not until after the line is completed and the carriage return key is struck that the compiler discovers the error and emits the error message "Identifier not delcared [sic]."

35. Since the Atkinson et al. incremental compiler is inevitably idle and can do no useful work while a line of source code is being typed in at the keyboard, the use of the IBM keyboard interrupt system, as in the Lawrence et al. or Maddock screen editing systems, would not enable the Atkinson et al. compiler to run any faster or improve the compiler performance in any way. Therefore the "exchange" proposed by the Examiner would be futile and pointless and would not be obvious to one skilled in the compiler art.

36. In Paragraph 22 of the Office Action the Examiner repeatedly relies upon assertions that in the Maddock patent allegedly:

(1) "The reformatting bursts of a line at a time are short compared to typical inter-stroke time..."

(2) "...the reformatting time for one line was less than the inter-keystroke time..."

(3) "...due to the differences in the reformatting time and inter-keystroke time..."

The Examiner has overlooked that these assertions and the arguments based thereon would no longer be tenable if the Atkinson et al. compiler were substituted for the Maddock

reformatter as proposed by the Examiner. Compilation of a line of source code in a high level language, even an emasculated student version of Pascal, is a relatively slow time-consuming process compared to the time between keystrokes. The lengthy time required for compilation is the very reason for the present invention. I estimate that the Atkinson compilation time for one line of Pascal code would be at least an order of magnitude greater than the inter-keystroke time of even a slow typist.

37. The main differences between the respective overall modes of operation of the references relied upon by the Examiner and the overall mode of operation of the system disclosed in the Reiffin application and defined in his claims may be summarized as follows:

(a) The Atkinson compiler is incremental, as described on Pages 3 and 4 of the Reiffin specification, in that the Atkinson compiler must wait until an entire line of source code is typed in before the compiler can do any processing, and during this waiting period the Atkinson compiler is idle and can do no useful work. On the other hand, the disclosed and claimed Reiffin compiler continuously processes the source code in the background between keystrokes as long as there are unprocessed lines in the editor buffer, except during each brief period for performing the editing operation required by each keystroke. The Atkinson compiler does not permit the programmer to type any additional keystrokes until the compilation of the previous line is completed. The disclosed and claimed Reiffin compiler enables the programmer to continue typing source code into the buffer, without waiting for the compiler, irrespective of the progress made by the compiler. As a result, the Atkinson compiler can

reduce (but not eliminate) the waiting and tedium involved in the conventional compiling method only by emasculating the language so as to simplify and speed up the lexical, syntactic and semantic analyses. This is why this incremental compiler approach is inherently limited to small languages for "novices" if the waiting periods between lines is to be brief enough to be tolerable.

(b) The IBM patent to Lawrence does not disclose either the mode of operation or the hardware or the software so as to enable a determination as to how it operates or how its operation compares with the operation disclosed in the subject Reiffin specification or recited in the Reiffin claims. Furthermore, the IBM patent to Lawrence discloses nothing which would indicate that its mode of operation is different from that of the IBM Personal Computer as described above and in the attached exhibits.

(c) The IBM patent to Maddock is also incremental in the sense that a complete line of code must be formatted before the editing operation can be invoked. This is acceptable because formatting, unlike compiling or other language analysis, is so fast that the formatting of a line is almost instantaneous as compared with the "interstroke time" of a typist. If Maddock performed compilation (or other language analysis) instead of mere formatting, then it would be merely a conventional incremental compiler with the disadvantages described on Pages 3 and 4 of the Reiffin specification.

38. For the reasons stated above, if I had read the Lawrence, Maddock and Atkinson references, without having read the Reiffin specification and drawings, it would not have been obvious to me to modify or combine these references in the manner

proposed by the Examiner in the Office Action. Since I have more than ordinary skill in the systems programming art I infer that the Examiner's proposed modification of these references would not have been obvious to one of ordinary skill in said art. This inference is supported by the fact that, insofar as I am aware, no one has ever combined systems similar to those disclosed in these references in the manner proposed by the Examiner, even though incremental compilers similar to the Atkinson compiler and formatting and editing terminals similar to those of Lawrence et al. and Maddock have been widely known for at least a decade.

/s/

Lawrence Oppenheim

Subscribed and sworn to before me this _____ day of September, 1989.

/s/

Notary Public